Rodziny zbiorów rozłącznych

Pewien zestaw elementów organizujemy w rodziny zbiorów rozłącznych. Każdy zbiór posiada jeden wyróżniony element, który jest *reprezentantem zbioru* zawierającego ten element.

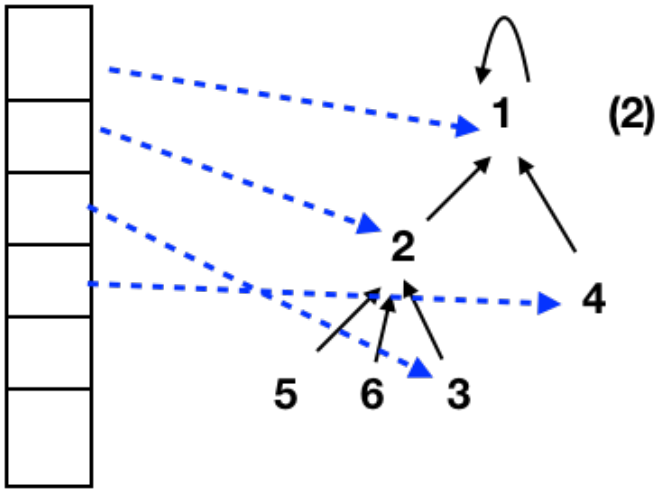
Operacje:

`MakeSet` – tworzy jednoelementowy zbiór

`FindSet(x)` – daje jako wynik reprezentanta zbioru, do którego należy `x`

`Union(x,y)` – łączy zbiory, których reprezentantami są `x` i `y` w jeden zbiór

Reprezentacja drzewiasta



Struktura węzła `x` :

`x.key` - klucz związany z elementem `x` (istotny tylko dla wydruku)
`x.p` - wskaźnik do ojca; korzeń wskazuje sam na siebie

Ponadto dla wersji z rangą

`x.rank` - ranga drzewa czyli oszacowanie wysokości, określone tylko dla korzenia

Operacje:

```
MakeSet(k)
// utworzenie zbioru reprezentującego jednoelementowy zbiór z kluczem k
   utworz węzeł x
   x.key = k
   x.p = x
   x.rank = 0 // tylko dla wersji z rangą
   return x

FindSet(x)
// zwraca reprezentanta zbioru do którego należy element (czyli węzeł) x
// Wersja bez kompresji: zwracany jest korzeń drzewa do którego należy x
   if x != x.p
      return FindSet(x.p)
   else
      return x

FindSet(x)
// zwraca reprezentanta zbioru do którego należy element (czyli węzeł) x
// Wersja z kompresją: każdy węzeł na ścieżce od "x" do korzenią
// jest doczepiany bezpośrednio do korzenia, czyli korzeń staje się
// jego ojcem
   if x != x.p
       x.p = FindSet(x.p)
   return x.p
```

```
Union(x,y)
// zamienia zbiory o reprezentantach x,y w jeden zbior będący ich sumą
// rozłączną Wersja bez rangi: "y" jest doczepiany do "x", czyli
// "x" staje się ojcem "y"
   y.p=x
```

```
Union(x,y)
// zamienia zbiory o reprezentantach x,y w jeden zbiór będący ich sumą
// rozłączną. Wersja z rangą: zbiór o mniejszej randze jest doczepiany
// do korzenia zbioru o większej randze; jeżeli rangi są równe to
// "y" do "x" i ranga "x" wzrasta o 1
   if x.rank < y.rank
      x.p = y
   else
      y.p = x
      if x.rank==y.rank
         y.rank = y.rank+1
```

Złożoność czasowa

Szacujemy pesymistyczną złożoność czasową ciągu m operacji `MakeSet`, `FindSet`, `Union`, w którym jest n operacji `MakeSet`.

Stwierdzenie 3. Dla reprezentacji drzewiastej bez rangi i bez kompresji ścieżek powyższa złożoność jest $\Theta(m\,n)$.

Dowód

`MakeSet(`v_1`)` ... `MakeSet(`v_n`)`

`Union(`v_2`,`v_1`)` $v_2 \leftarrow v_1$, drzewo o wysokości 1

`Union(`v_3`,`v_2`)` $v_3 \leftarrow v_2 \leftarrow v_1$, drzewo o wysokości 2

`Union(`v_4`,`v_3`)` $v_4 \leftarrow v_3 \leftarrow v_2 \leftarrow v_1$, drzewo o wysokości 3

.....

`Union(`v_n`,`v_{n-1}`)` $v_n \leftarrow v_{n-1} \leftarrow \ldots \leftarrow v_1$, drzewo o wysokości n-1

`Findset(`v_1`)` , czas $\Theta(n)$

`Findset(`v_1`)` , czas $\Theta(n)$

.....

Razem czas:

$$\Theta(n) + \Theta(n-1) + (m-2n-1)\Theta(n) = \Theta(mn)$$

jeżeli m będzie znacząco większe niż n

Stwierdzenie 4. Dla reprezentacji drzewiastej z rangą i bez kompresji ścieżek powyższa złożoność jest $O(m \lg n)$.

Dowód można sprawdzić, że przy łączeniu z rangą wysokość drzewa jest taka jak ranga i jest nie większa niż logarytm z ilości elementów w drzewie. Elementów jest w sumie n, więc złożoność każdej operacji `Findset` jest $O(\lg n)$, pozostałe operacje mają czas stały.

Stwierdzenie 5. Dla reprezentacji drzewiastej z rangą i z kompresją ścieżek powyższa złożoność jest $O(m \cdot \alpha(m, n))$, gdzie $\alpha(m, n)$ jest pewną bardzo wolno rosnącą funkcją. Dla wartości m, n spotykanych w praktyce można przyjąć, że $\alpha(m, n) \leq 5$, zatem powyższa złożoność to $O(m)$.